SCIENCES SORBONNE UNIVERSITÉ

2I002
EXERCICES D'APPLICATION

Vincent Guigue

LIP6

MLIA Machine Learning & Deep Learning for Information Access

Lapin (interro 2015)

La classe `Lapin` possède:

- deux attributs codant sa position double `posx` et `posy`, et un attribut int `nbEnfants` codant le nombre d'enfants du lapin.
- un constructeur à 2 arguments initialisant les deux attributs de position. `nbEnfants` est toujours initialisé à 0.
- un constructeur sans argument, qui initialise les attributs de position aléatoirement entre -10 et 10 avec `this()`.
- `toString` : génère une chaine de caractère de la forme: [*posx*, *posy*, nbEnfants=...]
- `deplacer` : prend en argument 2 doubles et déplace le lapin en les utilisant,
- `reproduire` : Le lapin créé est retourné par la méthode `reproduire` si les lapins sont proches, `null` sinon.
- `estParent` retourne un booléen qui vaut `true` si le lapin a déjà eu des enfants.

Soit les lignes de code suivantes (dans un main): combien y a-t-il de variables et combien d'instances?

```
1 Lapin l1 = new Lapin();
2 Lapin l2 = new Lapin();
```

```
3 Lapin l3 = l2;
4 l3.deplacer(2, 4);
```

Quelles lignes vous semblent correctes? Justifier les (4) erreurs en indiquant notamment si elles empêchent la compilation ou l'execution du programme. A quelle position se trouvent les lapins à la fin de ce programme? Quels sont les affichages dans la console?

```
1  Lapin l4 = new Lapin(1,2);
2  Lapin l5 = l4;
3  Lapin l6;
4  l6.deplacer(1, 2);
5  Lapin l7 = l6;
6  Lapin l7 = new Lapin(1,3);
7  Lapin bebeLapin = l4.reproduire(l7);
8  System.out.println(bebeLapin.toString());
9  l5.deplacer(2, 4);
10 l6.deplacer(1, 2);
11 System.out.printn("nb Enfants de L4: "+ l4.nbEnfants);
12 System.out.printn("L4 est-il parent? : "+ l4.estParent());
```

Complexes et composition

La classe `Complexe` possède:

- deux attributs double `reelle` et `imag`,
- un constructeur à 2 arguments initialisant les deux attributs
  NB: signature obligatoire: `public Complexe(double reelle, double imag)`,
- un constructeur sans argument, qui initialise les arguments aléatoirement entre -2 et 2 (avec `this()`).
- `toString` qui génère une chaine de caractère de la forme: (*reelle* + *imag* `i`)
- `addition` de deux complexes,
- `multiplication` de deux complexes,
- `estReel` qui teste si le complexe est en fait réel (dans le cas où la partie imaginaire est nulle).
- une méthode :

```
public void translateDeUn(){ reelle+= 1; imag += 1; }
```

- Donner le code de la classe `SegmentComplexe` qui contient 2 complexes en attributs. Ajouter un constructeur à deux arguments et des accesseurs.

```
1  Complexe c1 = new Complexe(1,0);
2  Complexe c2 = new Complexe(2,0);
3  Complexe c3;
4  SegmentComplexe s1 =
5     new SegmentComplexe(c1, c2);
6
7  c3 = c1.addition(c2);
8  System.out.println(c3);
9  if(c3.estReel())
10   System.out.println("c3 est reel");
11
12 System.out.println(s1);
13 c1.translateDeUn();
14 System.out.println(s1);
15 c1 = c3;
16 System.out.println(s1);
17
18 Complexe c4;
19 if(c4.estReel())
20   System.out.println("c4 est reel");
```

Ce code contient une erreur (qui empêche l'exécution): trouver là (après la ligne 10).

Donner les affichages produits lors de l'exécution (en imaginant que le problème précédent a été réglé).

Encore des vecteurs

Relever les erreurs et proposer des solutions pour corriger le code. A chaque affichage, donner ce qui s'affiche (ou ce qui devrait s'afficher une fois le code corrigé). Prédire si les clauses des *if* sont satisfaites ou non.

```
1  Vecteur v1 = new Vecteur(1,2);
2  Vecteur v2 = new Vecteur();
3  System.out.println(v1.toString()+"\n"+v2.toString());
4  Vecteur v3;    v3.x = 5;    v3.y = 7;
5  Vecteur v3 = new Vecteur(5,7);
6  double d = scal(v3);
7  double d = v1.scal(v3);
8  v1.translate(3,4);    v1.afficher();
9  Vecteur v4 = v1.addition(v3);
10 double s = v2.scal(v1);
11 if(s<30)  v1.afficher();
12 else      v2.afficher();
13 v1.x = v1.x + 10;       v1.traslate(10,0);
14 if(v1.getX() < 10 || v2.getX() < 10)
15         System.out.println("On passe ici!");
16 if(v2.getY() < 10 && v3.getX() <= 5)
17         System.out.println("On passe ici aussi!");
18 if(v2.getY()+5 < 10 && v3.getX() > 5 || s<30)
19         System.out.println("On passe ici aussi aussi!");
```